Evaluating MT systems with BEER

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 104 OCTOBER 2015 17–26

Evaluating MT systems with BEER

Miloš Stanojević, Khalil Sima'an

Institute for Logic, Language and Computation, University of Amsterdam

Abstract

We present BEER, an open source implementation of a machine translation evaluation metric. BEER is a metric trained for high correlation with human ranking by using learning-to-rank training methods. For evaluation of lexical accuracy it uses sub-word units (character n-grams) while for measuring word order it uses hierarchical representations based on PETs (permutation trees). During the last WMT metrics tasks, BEER has shown high correlation with human judgments both on the sentence and the corpus levels. In this paper we will show how BEER can be used for (i) full evaluation of MT output, (ii) isolated evaluation of word order and (iii) tuning MT systems.

1. Introduction

Machine Translation (MT) evaluation deals with the estimation of a measure (possibly distance) of the quality of some hypothesis MT output to given human translations, usually treated as gold standard translations. Often times, simplistic heuristics, such as counts of n-gram matches, are used. When the two corpora being compared are of relatively large size (>2000 sentences) the estimate can be reliable, even with simple measures such as BLEU (Papineni et al., 2002), because the collected sufficient statistic is reliable enough.

However, when we turn to evaluation at the sentence level, we cannot get away with such simple heuristic measures based on simple counting of too sparse statistics. We believe that evaluation should be treated as a modeling task (just like parsing or POS tagging or other NLP tasks) where we should train our models to learn the specific aspects of language processing, using a wide range of quality indicators (features). This is the motivation for making BEER a trained metric.

 Corresponding author: m.stanojevic@uva.nl
Cite as: Miloš Stanojević, Khalil Sima'an. Evaluating MT systems with BEER. The Prague Bulletin of Mathematical Linguistics No. 104, 2015, pp. 17–26. doi: 10.1515/pralin-2015-0010.

It is no surprise that trained metrics perform much better than heuristic metrics. BEER has been the best sentence level metric on WMT14 metrics task (Macháček and Bojar, 2014), and one of the best on both corpus and sentence level on WMT15 metrics task (Stanojević et al., 2015a). On the WMT15 tuning task for Czech-English BEER was the best submitted system falling behind only over the strong baseline (Stanojević et al., 2015b).

Unfortunately, trained metrics are often not that easy to use. Furthermore, these metrics are mostly made for the metrics tasks and often not published online, and when they are published online, this is done without the trained models or without suitable documentation. With this paper we aim to document BEER as a trained metric that performs well but also is easy to use, offering a range of attractive properties that researchers are used to see in the simple measures (for example statistical testing, tuning and many more).

In this paper we will concentrate only on the usage of BEER, but BEER has many interesting aspects that are presented elsewhere:

- evaluation of word order using permutation trees (PETs) (Stanojević and Sima'an, 2014b)
- character n-gram matching (Stanojević and Sima'an, 2014a)
- a learning-to-rank model (Stanojević and Sima'an, 2014a)
- a corpus level score that decomposes to sentence level scores (Stanojević and Sima'an, 2015)
- a tuning model that is not biased for recall (Stanojević and Sima'an, 2015)
- a Treepel version based on syntactic features (Stanojević and Sima'an, 2015)

In the next section we will briefly summarize some these aspects.

2. BEER basics

The model underying the BEER metric is flexible for the integration of an arbitrary number of new features and has a training method that is targeted for producing good rankings among systems. Two other characteristic properties of BEER are its hierarchical reordering component and char n-grams lexical matching component.

BEER is essentially a linear model with which the score can be computed in the following way:

$$score(h, r) = \sum_i w_i \times \phi_i(h, r) = \vec{w} \cdot \vec{\phi} \quad (1)$$

where $\vec{w}$ is a weight vector and $\vec{\phi}$ is a feature vector.

2.1. Learning-to-rank

Since the task on which our model is going to be evaluated is ranking translations it comes natural to train the model using *learning-to-rank* techniques.

Our training data consists of pairs of "good" and "bad" translations. By using a feature vector $\vec{\phi}_{good}$ for a good translation and a feature vector $\vec{\phi}_{bad}$ for a bad translation then using the following equations we can transform the ranking problem into a binary classification problem (Herbrich et al., 1999):

$$\begin{aligned} score(h_{good}, r) > score(h_{bad}, r) \Leftrightarrow \\ \vec{w} \cdot \vec{\phi}_{good} > \vec{w} \cdot \vec{\phi}_{bad} \Leftrightarrow \\ \vec{w} \cdot \vec{\phi}_{good} - \vec{w} \cdot \vec{\phi}_{bad} > 0 \Leftrightarrow \\ \vec{w} \cdot (\vec{\phi}_{good} - \vec{\phi}_{bad}) > 0 \\ \vec{w} \cdot (\vec{\phi}_{bad} - \vec{\phi}_{good}) < 0 \end{aligned} \tag{2}$$

If we look at $\vec{\phi}_{good} - \vec{\phi}_{bad}$ as a positive training instance and at $\vec{\phi}_{bad} - \vec{\phi}_{good}$ as a negative training instance, we can train any linear classifier to find weight the vector $\vec{w}$ that minimizes mistakes in ranking on the training set.

In practice BEER uses logistic regression as implemented in Weka toolkit (Hall et al., 2009). So the estimated weights are used in the following way:

$$score(h, r) = \frac{2}{1 + e^{-\sum_i w_i \times (\phi_i(h,r) - \phi_i(r,r))}} \tag{3}$$

The main difference from Equation1 is that here:

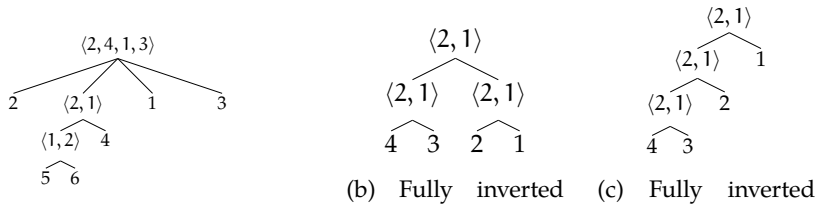
1. first substract features of system translation given reference and reference given reference
2. apply sigmoid function and then
3. we multiply with 2

This formula does not make a difference in ranking compared to Equation 1 but it makes a difference in scaling the scores. Motivation for it is explained in Stanojević and Sima'an (2015). This scaling is important for getting a better corpus level score that is calculated as average sentence level score over whole corpus:

$$BEER_{corpus}(c) = \frac{\sum_{s_i \in c} BEER_{sent}(s_i)}{|c|} \tag{4}$$

2.2. Lexical component based on char n-grams

Lexical scoring of BEER relies heavily on character n-grams. Precision, Recall and F1-score are used with char n-gram orders from 1 until 6. These scores are more smooth on the sentence level than word n-gram matching that is present in other metrics like BLEU (Papineni et al., 2002) or METEOR (Michael Denkowski and Alon Lavie, 2014).



(a) Complex PET

(b) Fully inverted PET 1

(c) Fully inverted PET 2

Figure 1: Examples of PETs

BEER also uses precision, recall and F1-score on word level (but not with word n-grams). Matching of words is computed over METEOR alignments that use WordNet, paraphrasing and stemming to have more accurate alignment.

We also make distinction between function and content words. For more precise description on used features and their effectiveness you can look at Stanojević and Sima'an (2014a).

2.3. Reordering component based on PETs

Alignment between system and reference translation can be simplified and considered as permutation of words from the reference translation in the system translation. Previous work by Isozaki et al. (2010) and Birch and Osborne (2010) used this permutation view of word order and applied Kendall τ for evaluating its distance from ideal (monotone) word order.

BEER goes beyond this *skip-gram* based evaluation and decomposes permutation into a hierarchical structure which shows how subparts of permutation form small groups that can be reordered all together. Figure 1a shows PET for permutation $\langle 2, 5, 6, 4, 1, 3\rangle$. Ideally the permutation tree will be filled with nodes $\langle 1, 2\rangle$ which would say that there is no need to do any reordering (everything is in the right place). BEER has features that compute the number of different node types and for each different type it assigns a different weight. Sometimes there are more than one PET for the same permutation. Consider Figure 1b and 1c which are just 2 out of 3 possible PETs for permutation $\langle 4, 3, 2, 1\rangle$. Counting the number of trees that could be built is also a good indicator of the permutation quality.

3. Installing BEER

BEER is implemented in Scala so the only requirement for running it is just having the latest version of Java virtual machine installed (at least version 8). All the dependencies of BEER are included with the installation except METEOR and Stanford

CORE dependency parser (Chen and Manning, 2014) which gets installed automatically the first time BEER is ran.

The basic procedure to install BEER is with the following commands in terminal:

```
wget https://staff.fnwi.uva.nl/m.stanojevic/beer/beer_1.1.tar.gz
tar xfvz beer_1.1.tar.gz
./beer_1.1/beer  # this installs METEOR and Stanford parser
rm beer_1.1.tar.gz
```

4. Usage from command line

BEER has several *working modes*. They specify if we want to use BEER for evaluation, for computing features, for training, for evaluating reordering or we want to use it for interactive evaluation on the terminal. We will explain three modes that are most useful from these: evaluation, evaluateReordering and interactive mode.

Bellow is an example of using BEER with evaluation working mode, where system translation is given with parameter `-s`, reference translation with parameter `-r` and language with parameter `-l`.

```
./beer --workingMode evaluation -l en -s system.en -r reference.en
```

Because `--workingMode evaluation` is the default setting we can also skip that parameter and just write:

```
./beer -l en -s system.en -r reference.en
```

This command will print the corpus level BEER score. To get the sentence level scores we just need to add `--printSentScores` to the command. The language parameter `-l` is an obligatory parameter for BEER because BEER uses language specific models for scoring and language specific resources (parsers, function words lists, paraphrase tables, stemmers...) for aligning reference and system translation. All languages from WMT13 and WMT14 are supported at this point. There is additional language *other* which is recommended in case there is no language specific model available.

This and some other parameters of BEER are shown in Table 1.

5. Usage in interactive mode

In some use cases the user might want to connect some other application with BEER. This can be done by using BEER as a library in case the other application executes on Java virtual machine, but the more general solution is usage of BEER through the interactive command line. This allows usage of BEER from any application which can read and write through pipe to some program. Here we describe this interactive way of using BEER.

parameter	useage
`-l`	input language
`-s`	system translation
`-r`	reference translations separated by column
`--printSentScores`	pritns BEER score for each sentence
`--printFeatureValues`	prints feature values for each sentence
`--norm`	tokenizes the input using METEOR tokenizer
`--noLower`	stops BEER from lowercasing the input
`--noPunct`	excludes punctuation from evaluation
`--help`	prints these and some other parameters of BEER

Table 1: Command line parameters of BEER

To start the interactive shell we need to set the working mode and the language for evaluation:

```
./beer --workingMode interactive -l other
```

When the interactive shell starts, we can type different commands for evaluation. To evaluate for a sentence level score we can type the following:

```
EVAL ||| system translation ||| reference 1 ||| reference 2
```

and then as output we should get the score for each reference translation. If we want only the best score out of all references we just need to type `EVAL BEST` instead of `EVAL`. In case we need feature values, the format is the same but we use `FACTORS` command instead of `EVAL`. Finally, to exit it is enough to type `EXIT`.

6. Statistical testing of BEER scores using MultEval

Usually it is not enough to know the final score of the system and whether it is better (or worse) than the baseline but also to know whether this difference is statistically significant. The tool that became quite popular for this task in MT community if Multeval (Clark et al., 2011) that has support for BLEU, TER and METEOR. With BEER we distribute the version of Multeval that contains a metrics module for BEER with the same interface as for the other metrics.

Here we describe how to use BEER with Multeval. Current implementation has no bugs that we are aware of, but it is relatively slow (it requires running in parallel and relatively large amount of RAM memory). This is likely to change soon.

Here is the command for running Multeval with BEER:

```
./multeval eval --metrics beer \
                --beer.language en \
                --refs references.en \
                --hyps-baseline translations.en.*
```

Basically the only additional obligatory parameter is `--beer.language`, but other than that all other parameters are standard Multeval parameters.

7. Tuning Moses for BEER – beta

BEER has support for tuning Moses (Koehn et al., 2007) systems parameters for higher BEER score. In principle all Moses optimization algorithms could be used, but we tested it only with *kbmira* (Cherry and Foster, 2012). For now, in order to add support for tuning Moses with BEER the user needs to recompile Moses by first adding files located in *src_moses* of BEER installation into Moses directory and then compiling. In future releases we hope to add support for BEER in the standard Moses installation so this manual compilation would not be necessary.

When Moses is compiled with the necessary C++ files, tuning can be done in the same way as usual by calling *mert-moses.pl* and by specifying BEER parameters in `--batch-mira-args` in the following way:

```
perl $MOSES/scripts/training/mert-moses.pl \
  --batch-mira-args="--sctype BEER --scconfig beerDir:$BEER_DIR,
  beerLang:en,beerModelType:tuning,beerThreads:30" ...
```

With *beerDir* we specify where Moses can find the installation of BEER that it can use. Standard models that are trained for human correlation have heavy recall bias and they are not good for MT tuning. That is why BEER has models without this bias that perform much better for tuning (Stanojević and Sima'an, 2015). To specify that we want to use that kind of model we use parameter *beerModelType* (currently only English is supported) and *beerLang* tells which language is evaluated.

8. Evaluating word order with PETs

BEER is a full evaluation metric that scores translation by all ascpects (both fluency and adequacy). But sometimes we want to put more attention on the evaluation of ether adequacy or fluency. Evaluating adequacy can be pretty streightforward, which is not the case for evaluating fluency. Metrics that are more successful in fluency treat this problem as measuring distance between the permutation of words in system translation from the ideal permutation (Birch and Osborne, 2010; Isozaki et al., 2010). In Stanojević and Sima'an (2014b) evaluation over permutations was extended from

function	origin
Kendall	Birch and Osborne (2010); Isozaki et al. (2010)
Spearman	Isozaki et al. (2010)
Ulam	Birch et al. (2010)
Hamming	Birch and Osborne (2010)
Fuzzy	Talbot et al. (2011)
PETrecursiveViterbi	Stanojević and Sima'an (2014b)
PEFrecursive	Stanojević and Sima'an (2014b)

Table 2: Implemented $\texttt{ordering}(\cdot,\cdot)$ functions

treating permutations as flat sequential structures to the hierarchical structures that better explain the reordering patterns.

In BEER installation, appart from standard trained BEER linear models, there is additionally an implmentation of the following interpolation of lexical and ordering score:

$$\texttt{score}(s,r) = \alpha\, F_1(s,r) + (1-\alpha)\, \texttt{ordering}(s,r) \tag{5}$$

F_1 is the lexical measure over unigrams and $\texttt{ordering}(\cdot,\cdot)$ is an ordering function over alignments (permutation) between words from system and reference translation. There are many implemented $\texttt{ordering}(\cdot,\cdot)$ functions shown in Table 2.

We can do the scoring with the following command:

```
./beer --workingMode evaluateReordering \
       --alpha 0.5 \
       --reorderingMetric PEFrecursive \
       -l en -s system.en -r reference.en
```

Here α parameter is specified with `--alpha` and $\texttt{ordering}(\cdot,\cdot)$ function with `--reorderingMetric`. Other BEER parameters are mostly the same.

9. Summary

We have presented different ways in which BEER software can be used for evaluation and optimization of MT systems. We hope that this software package would increase usage of tunable metrics with state-of-the-art correlation with human judgment over standard metrics that are based on heuristics and usually perform badly on corpus and especially sentence level. BEER is licensed under GPL license and is available at `https://github.com/stanojevic/beer`.

Acknowledgements

This work is supported by STW grant nr. 12271 and NWO VICI grant nr. 277-89-002.

Bibliography

Birch, Alexandra and Miles Osborne. LRscore for Evaluating Lexical and Reordering Quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1749.

Birch, A., M. Osborne, and P. Blunsom. Metrics for MT evaluation: evaluating reordering. *Machine Translation*, pages 1–12, 2010. ISSN 0922-6567.

Chen, Danqi and Christopher D Manning. A Fast and Accurate Dependency Parser using Neural Networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Cherry, Colin and George Foster. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Clark, Jonathan H., Chris Dyer, Alon Lavie, and Noah A. Smith. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002774.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278.

Herbrich, Ralf, Thore Graepel, and Klaus Obermayer. Support Vector Learning for Ordinal Regression. In *In International Conference on Artificial Neural Networks*, pages 97–102, 1999.

Isozaki, Hideki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1870658.1870750.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

Macháček, Matouš and Ondřej Bojar. Results of the WMT14 Metrics Shared Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland,

USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3336.

Michael Denkowski and Alon Lavie. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the ACL 2014 Workshop on Statistical Machine Translation*, 2014.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://dx.doi.org/10.3115/1073083.1073135.

Stanojević, Miloš and Khalil Sima'an. Fitting Sentence Level Translation Evaluation with Many Dense Features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 202–206, Doha, Qatar, October 2014a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1025.

Stanojević, Miloš and Khalil Sima'an. Evaluating Word Order Recursively over Permutation-Forests. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 138–147, Doha, Qatar, October 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4017.

Stanojević, Miloš and Khalil Sima'an. BEER 1.1: ILLC UvA submission to metrics and tuning task . In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, June 2015. Association for Computational Linguistics.

Stanojević, Miloš, Amir Kamran, and Ondřej Bojar. Results of the WMT15 Metrics Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, June 2015a. Association for Computational Linguistics.

Stanojević, Miloš, Amir Kamran, and Ondřej Bojar. Results of the WMT15 Tuning Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, June 2015b. Association for Computational Linguistics.

Talbot, David, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz J. Och. A Lightweight Evaluation Framework for Machine Translation Reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 12–21, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-12-1. URL http://dl.acm.org/citation.cfm?id=2132960.2132963.

Address for correspondence:
Miloš Stanojević
m.stanojevic@uva.nl
P.O. Box 94242 , Amsterdam, The Netherlands